УДК 004.891

ТЕСТИРУЮЩИЙ КОМПЛЕКС «ВЫБОР БУДУЩЕЙ ПРОФЕССИИ»

И. В. КАТУШЁНОК, Д. И. МАКЕЁНОК
(Представлено: канд. техн. наук, доц. В. М. ЧЕРТКОВ)

В статье рассматривается процесс разработки приложения для прохождения тестов по профориентации, созданного с использованием технологии Windows Presentation Foundation. Цель приложения – предоставление детям и подросткам удобного и интуитивно понятного инструмента для выявления их профессиональных предпочтений и способностей. В статье рассматриваются основные этапы проектирования и реализации приложения, включая архитектуру, выбор технологий и пользовательский интерфейс.

Введение. Профориентационное тестирование – это важный инструмент для определения профессиональных качеств, способностей и интересов человека. С его помощью можно выявить наиболее подходящие сферы деятельности, соответствующие личным качествам и предпочтениям. Профориентация помогает выбрать правильное направление для учебы или карьеры. В процессе тестирования используются различные методы и тесты, которые оценивают навыки, интересы и ценности. Это даёт более полное представление о том, какие профессии могут быть наиболее успешными и удовлетворяющими для конкретного человека.

Тестирующий комплекс предназначен для того, чтобы помочь людям определить свои профессиональные склонности и интересы. Он включает в себя ряд тестов и методик, которые позволяют выявить сильные и слабые стороны, а также наиболее подходящие сферы деятельности для каждого участника. Также важно отметить, что данная система относиться к области искусственного интеллекта, потому как она принимает решение основываясь на ответах пользователей.

Программная часть. Начать следует с выбора инструментов разработки, а именно среды разработки и языка программирования. Одной из самых популярных сред разработки приложений под Windows является Visual Studio. Visual Studio является крайне ресурсозатратной средой разработки, что может отражаться в долгих загрузках проектов и требованием большого количества оперативной памяти. Главными плюсами Visual Studio являются: доступность всех необходимых инструментов разработки, возможность установки недостающих компонентов, что позволяет легко и доступно вести разработку различных проектов.

Если говорить о выборе языка программирования, то отличным выбором является C#. C# – это современный объектно-ориентированный язык программирования. C# прекрасно работает с различными технологиями Microsoft для разработки приложений с пользовательским интерфейсом, такими как WPF, WinForms и UWP, что позволяет разрабатывать приложения для разнообразных целей и устройств. C# активно поддерживается компанией Microsoft и имеет огромное сообщество разработчиков.

Технологией разработки приложения является уже упомянутая Windows Presentation Foundation (WPF). WPF имеет множества различных достоинств для разработки приложений с графическим интерфейсом.

WPF позволяет разрабатывать приложения, используя как разметку, так и код программной части, что привычно для разработчиков на ASP.NET. Разметка XAML обычно используется для определения внешнего вида приложения, а управляемые языки программирования – для реализации его поведения [1].

Основными причинами использования WPF для разработки приложений под Windows являются:

– WPF предоставляет широкий набор элементов управления, что упрощает разработку сложных пользовательских интерфейсов.

– возможность легко изменять внешний вид приложения через стили и шаблоны, делая его адаптируемым к различным визуальным требованиям и предпочтениям пользователей.

– WPF является проверенной временем и хорошо зарекомендовавшей себя технологией, поддерживаемой Microsoft. Это обеспечивает её стабильность и долговечность, а также большую информационную базу.

После выбора технологий разработки, стоит уделить внимание архитектурному решению проекта. В современном программировании архитектуры и паттерны играют очень важную роль в разработке. Они позволяют структурировать проект, сделать его более понятным для разработчиков и ввести некий шаблон, на основе которого будет строиться проект. Также архитектуры представляют собой решение определённых проблем с масштабируемостью и поддерживаемостью проекта. Самой распространённой архитектурой технологии WPF является Model-View-ViewModel (MVVM).

MVVM (Model-View-ViewModel) – это архитектурный шаблон, используемый в разработке программного обеспечения для разделения пользовательского интерфейса от бизнес-логики и данных [2].

MVVM предполагает наличие трех компонентов для каждого элемента приложения. Первый компонентом является модель (Model), она представляет данные компонента и также может содержать его логику. Вторым компонентом является представление (View). Представление – это графическая реализация элемента, то что он должен показывать и то как с ним нужно взаимодействовать. Последним компонентом является модель представление (ViewModel). Основное назначение модели представления связать данные из модели с представлением. Пример архитектуры приложения представлен на рисунке 1.1.

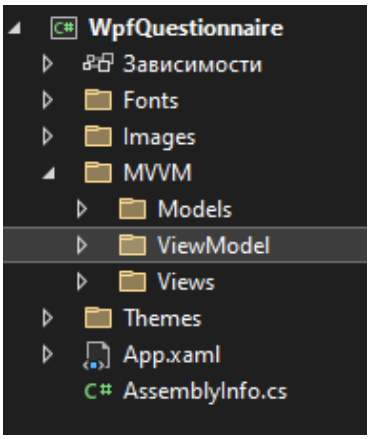

Рисунок 1.1. – Архитектура приложения

Теперь, когда были выбраны технология и архитектура можно преступить к более конкретной реализации. Для начала стоит выделить основные классы тестов – это вопрос и результат теста. Поскольку система результатов для двух тестов разная, классы ответов для них будут отличаться. Примеры классов теста и результатов показаны в листингах 1.1, 1.2 и 1.3.

Листинг 1.1. – класс Question

```
public class Question
{
private Dictionary<string, int> _scoreDictionary = new Dictionary<string, int>();
    public string QuestionString { get; set; }
    public string BackgroundSource { get; set; }
    public List<string> Answers { get; set; } = new List<string>();
    public List<int> AnswersScore { get; set; } = new List<int>();
    public void InitializeScoreDictionary()
    {
        for (int i = 0; i < Answers.Count; i++)
   {
            _scoreDictionary.Add(Answers[i], AnswersScore[i]);
        }
    }

    public int GetAnswersScore(string answer)
    {
        return _scoreDictionary[answer];
    }
}
```

Листинг 1.2. – класс ChildResult

```
public class ChildResult {
    public string ResultMessage { get; set; }
    public string Header { get; set; }
    public string BackgroundSource { get; set; }
    public int Index { get; set; }
}
```

Листинг 1.3. – класс TeenResult

```
public class TeenResult {
   public string ResultMessage { get; set; }
   public string Header { get; set; }
   public string BackgroundSource { get; set; }
   public int MinScore { get; set; }
   public int MaxScore { get; set; }
}
```

Информацию о вопросах и ответах будут храниться в формате json в директории приложения. Для хранения данных обычно используются базы данных, но учитывая фиксированное и небольшое количество используемых данных, в базе данных нет необходимости. Для взаимодействия с объектами из директории приложения, также реализован класс для загрузки вопросов и ответов.

После реализации классов моделей и класса для загрузки файлов из директории, стоит рассмотреть представления. С точки зрения представлений зрения представлений вопросы представляют из себя текст в виде вопроса, реализованный элементом TextBlock и несколько элементов RadioButton необходимых для выбора ответа. В представлении результата нужен только элемент TextBlock описывающий результат пройденного теста.

Осталось только объединить представления с моделями с помощью классов моделей представлений. Также классы моделей представлений содержат в себе логику для подсчёта результатов и выбора результата на основе количества набранных баллов.

Завершающим этапом разработки является разработка стилей. Стили помогают создать согласованность во всем приложении. Позволяют пользователям чувствовать себя комфортно, ведь им интуитивно понятно, где найти нужную информацию.

Заключение. В заключении можно сказать, что выбор WPF обеспечил гибкость и богатый набор элементов управления, способствующий созданию современного интерфейса, а применение MVVM позволило четко разделить бизнес-логику и логику интерфейса, улучшив модульность и тестируемость приложения. Достигнутые результаты подтверждают, что данная комбинация технологий является надежным выбором для реализации сложных приложений.

ЛИТЕРАТУРА

1. Learn Microsoft. Руководство по классическим приложениям (WPF .NET). [Электронный ресурс] Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-8.0 - Дата доступа: 11.10.2024.
2. AppTractor. Что такое MVVM архитектура. [Электронный ресурс] Режим доступа: https://apptractor.ru/info/articles/chto-takoe-mvvm-arhitektura.html - Дата доступа: 13.10.2024.
3. Cyberleninka. О разработке автоматизированной системы выбора направления будущей профессиональной деятельности. [Электронный ресурс] Режим доступа: https://cyberleninka.ru/article/n/o-razrabotke-avtomatizirovannoy-sistemy-vybora-napravleniya-buduschey-professionalnoy-deyatelnosti/viewer - Дата доступа: 14.10.2024